Introducing a Scalable Encryption Layer to Address Privacy and Security Issues in Hybrid Cloud Environments

Paul Reinhold and Wolfgang Benn
Chemnitz University of Technology
Chemnitz, Germany
Email: {paul.reinhold@s2012, wolfgang.benn@informatik}.tu-chemnitz.de

Benjamin Krause and Frank Goetz
Qualitype GmbH
Quality Management Systems
Dresden, Germany
Email: {b.krause,f.goetz}@qualitype.de

Dirk Labudde
Hochschule Mittweida
University of Applied Sciences
Mittweida, Germany
Email: dirk.labudde@hs-mittweida.de

***Abstract*—Besides security and privacy concerns, high efforts, necessary for developing and maintaining cloud services in the modern IT landscapes, are new major issues in small and medium enterprises. Software service development and operations face new challenges in dynamic cloud environments. To establish the cloud service and to address privacy and security issues, our suggested scalable Encryption Layer can be used. With our approach, small and medium enterprises, can securely outsource their data to public cloud storages preventing the public cloud provider from data insight, even with full access to the physical machines. This paper introduces an in-depth description for setting up our Encryption Layer, and also provides solutions for problems which may arise during the implementation and setup process of cloud environments. The presented test results of our implemented prototype demonstrate, with an overall overhead of 50% to 75%, the practical applicability.**

Keywords-Hybrid Cloud; Cloud Security; Architecture Layer; Industry Research; Small Medium Enterprises.

I. INTRODUCTION

The increasing knowledge about cloud computing technology and its publicity leads to a growing number of service offerings over the Internet. Even small and medium sized enterprises (SME) are able to offer services for a large number of consumers through cloud computing concepts. Like previous work [1] shows, with the usage of an appropriate hybrid cloud concept, SME can develop practical cloud service solutions with respect to privacy and security.

The high acceptance of services, like Instagram [2] or Dropbox [3] for private usage, suggests that private consumers have a lower privacy demand than business users. Studies of Gens, like [4], [5], support this hypotheses. A recent study of BITCOM [6] indicated that in Europe, and especially in Germany, the acceptance of public cloud services for business purposes is low. Typical reasons are security and privacy concerns. The most recent study of Crisp Research [7] has led to the same results. This study is most interesting for our work, since it mainly focused on SMEs. The study showed that around 40% of the respondents hesitate with cloud solutions, because their customers have concerns about security and privacy. Interestingly, with a percentage of around 60%, the main arguments against cloud computing solutions are the following: First, the high effort to run a cloud service and second, the high costs for developing a new cloud application. This shows, despite the security and privacy issues, SME hesitate because of deficient resources, like hardware, men power or know-how. Therefore, we aim to lower these concerns by introducing a convenient additional architecture layer, called *Encryption Layer* (EL). In a previous work [1] we demonstrate the practicability and low migration effort of this EL into existing software systems. In this paper we focus on a detailed insight of the implementation process and the effort to set up the appropriate environments. Our suggested EL is located between business logic and persistence layer. For evaluation purposes, we implement a prototype using the suggested architecture to outsource unstructured (files) and structured (databases) data into a public cloud in an encrypted and secured manner. For that purpose, our work focuses on SME providers that already run services or web applications and take new cloud offerings into account in an effort to become more cost-efficient, or to establish new business models, as discussed in [7]. In addition, SME providers probably use own hardware to run their services and plan to develop a new version or new service, which would exceed the current limit of their hardware. Another scenario is that the provider needs to invest in new hardware to keep its services running and is looking for lower cost alternatives.

The outline for the rest of the paper is as follows: In Section II, a comparison of common cloud delivery models with respect to privacy, costs and performance is discussed. Based on this comparison we elucidate the possible solutions with a reasonable effort for SME. In Section III, an abstract overview of our solution, as well as some differences to other hybrid cloud approaches is given. In addition, we discuss the applicability of a key management system inside of hybrid cloud environments. As the main part of the paper, Section IV summarizes the EL prototype implementation, and gives in-depth technical insights, as well as an evaluation of tests scenarios and results. Especially, we describe some pitfalls and typical problems, which may arise during the development process and appropriate solutions to avoid them. In Section V, a critical discussion of the test results is provided, as well as pros and cons of an additional layer, like EL, in general. In Section VII, we elucidate related and future works. Finally, Section VIII concludes the paper.

TABLE I. COMPARISON OF DIFFERENT CLOUD MODELS FROM CONSUMERS AND PROVIDERS POINT OF VIEW

View	Criteria	Private Cloud	Public Cloud	Hybrid Cloud
consumer	cost	high	low	medium
	privacy	medium	low	high
	data-at-rest encryption	yes	yes	yes
	key owner	provider	provider	consumer (and provider)
	key management	by provider	by provider	by provider (and consumer)
provider	cost	very high	low	medium
	availability	medium	very high	high
	backup	medium	very high	very high
	hardware needs	high	very low	medium
	effort to run service	very high	low	medium - high
	flexibility	high, but limited	very high	higher, but limited
	scaling	yes, but limited	yes	yes, but limited

II. CLOUD DEPLOYMENT MODEL COMPARISON

In the following sections, the end-user of a cloud service shall be named cloud consumer or simply consumer [8]. From the consumers view, the provider offers services over the Internet. Whether the service is offered by provider's hardware or by third party resources is irrelevant for the customer, as long as service supply is ensured. However, the method of providing can be essential for the acceptance of the service on consumers side.

According to literature [9], [10], [11], [12] most common cloud deployment models are private, public and hybrid cloud models. In the private cloud, the provider runs its own cloud. As Rhoton and Haukioja [13] mentioned some would argue that anything less than a full cloud model is not cloud computing. Actually, private cloud computing contradicts the idea of cloud computing through limitations in basic cloud characteristics defined by the National Institute of Standards and Technology (NIST) in [9] like rapid elasticity, on-demand self-service and resource pooling. Nevertheless, the term is widely accepted in academia and industry. Private cloud providers own the hardware and have exclusive access to it. To leverage cloud effects the provider runs its hardware in form of a cloud, allowing flexibility and scaling. Public clouds are offered by public cloud service providers (CSP). In contrast to private clouds, this form uses all advantages of cloud computing. Therefore, public clouds are the most flexible and cost-efficient services, since resources are obtained by need and payed by usage. As Fernandes et al. [14] mentioned, this model is less secure and more risky than other deployment models. Actually, this statement is trivial to confirm. Since public cloud computing is an extensive form of IT outsourcing, there is a much higher potential for malicious attacks compared to private cloud solutions. Including not only external attacks but internal attacks, e.g., through malicious administrators as well.

Hybrid clouds are the third common approach where services run both in a private and public clouds. We think this is the most interesting approach, having a high potential for balancing cloud computing advantages versus security and privacy issues. In addition, this approach seem the most likely one for using cloud computing in SME, van Hoeck et al. [15] supports this hypothesis. In our opinion the main aspect is to use private (expensive) resources as little as needed and public (cheap) resources as much as possible. However, this optimization is a highly complex task, especially for existing enterprise IT infrastructures or software systems.

Table I shows a comparison between the three cloud deployment models from both consumer's and provider's point of view. An actual survey on cloud security carried out by Fernandes et al. [14] has shown similar results. The costs factor for both consumer and provider is comprehensible, since hardware expenses are usually passed to consumers.

Privacy is low for public and medium for private cloud architecture. Authors, like Wang and Jia [16], argued that the private cloud could provide the the highest degree of security for users data. We do not fully agree with that, because this depends on who is the owner of the cloud and whose data is processed or stored in this private cloud. If data and cloud owner are the same, Wang and Jias [16] argument might be right. However, in our scenario, as well as in most cloud service offerings, the private cloud owner differs form the data owner. Private clouds often have strong authorization and access control concepts, but no special requirement to secure data with encryption against the provider (SME) itself [17]. Thus, the private cloud provider often has access to customer data and their customers data, respectively. Therefore, the argument of Wang and Jia [16] is not true in our scenario. In a public cloud it is very costly or impractical to secure data and to keep them available for processing at the same time, which, for instance, fully homomorphic encryption [18] can provide. However, approaches like Mylar [19] can pose an alternative, by the use of encryption in client software. Nevertheless, this cause additional tasks, like key management, suggesting a hybrid cloud approach offers a more attractive solution with respect to the customer demands. Another important aspect for consumers is data-at-rest encryption. This form of encryption is possible in all of the models, but implies tasks for key ownership and key management. If the same instance encrypts data and stores the referring key, no trustable security can be guaranteed, because providers can decrypt data without consumers knowledge or permission. This has been recently documented for economically rational cloud providers [20]. Because of this circumstance hybrid clouds suggest a solution where consumers get more control over their data and the possibility for public cloud providers to access unencrypted files is eliminated. Even if a consumer trusts its provider (with a private cloud) and consequently encryption is not needed, the hybrid solution is more economical. From the providers point of view, a private cloud can not provide the availability as it is guaranteed by a public cloud. The hybrid model benefits from this fact by outsourcing parts of the software

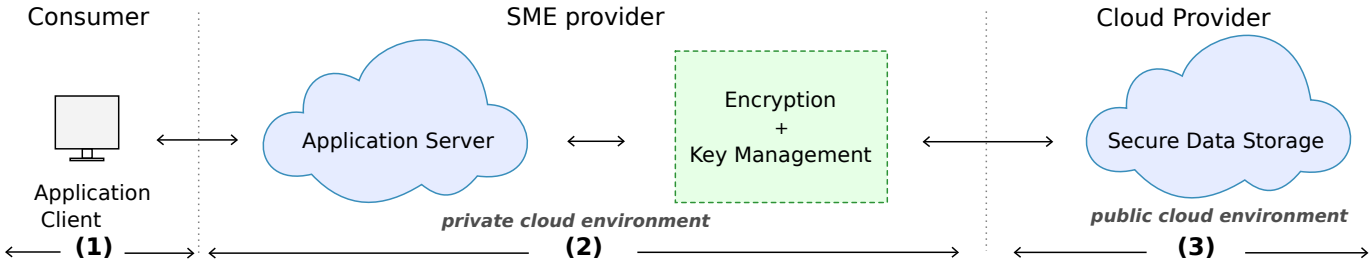


Figure 1. The hybrid cloud environment with Application Client, Application Server and outsourced Data Storage. The green dashed box is our additional Encryption Layer, located between the application server in the private cloud and data storage in the public cloud.

solution in a public cloud. Backup security underlays the same principle, in fact the backup process in the hybrid model can be outsourced completely. The hardware needs and the effort to run the service are coherent. Lots of own hardware means not only to manage, but also to maintain and have environment settings (buildings, redundant broadband internet access) to run a private cloud. As mentioned above, the great advantages of cloud computing like flexibility and scaling are limited in private and hybrid cloud solutions. As a result of this comparison and questions, our aim in [1] was to combine the security of a private cloud with the flexibility, reliability and availability of a public cloud, creating a balanced solution.

III. Hybrid Cloud Environments

The constellation of a hybrid cloud computing environment is illustrated in Figure 1. It consists typically of a client (1), a private (2) and a public cloud component (3). The most common usage of a hybrid cloud environment in literature [21], [22], [16], [23], [24] is a separation of sensitive and non-sensitive data, stored in private and public cloud, respectively. Lot of research has been done for developing efficient classification algorithms to separate or sanitize sensitive data from non-sensitive data. The aim is to outsource as much data as possible in public cloud. Our approach differs from this kind of hybrid cloud computing. By the use of encryption methods we outsource all kind of data, regardless their sensitivity. Resulting advantages are: no need for data labeling/classification, less effort to integrate in existing systems, and a less complex and therefore less error-prone overall system architecture.

A. Private cloud environment

As shown in Figure 1 the private cloud environment consists of application server and is managed by the SME provider. That means all business logic remains inside the private cloud, resulting in the main disadvantage of our hybrid cloud solution. Actually, there is no alternative if neither consumer nor SME provider trust a public cloud provider anyway. Since efficient data-in-use encryption is still an open issue; even with great improvements like fully homomorphic encryption [18]. Consequently, our approach uses the existing system of the SME provider, adding an additional architecture layer (green dashed box) for protecting data in the public cloud with encryption methods.

B. Public cloud environemt

The public cloud component is used for data storage in form of virtual servers provided by an public IaaS provider. As Rhoton [13] mentioned, this is the most basic form of using cloud computing resources. We do not consider storage solutions like S3 [25] or Azure SQL Database [26] to avoid vendor lock-in effects and be more flexible. However, as we address SME with this work the establishment of a basic, yet efficient cloud solution (as a first step) is our focus. As the cloud provider selection shows, there is no reason to use one public cloud provider exclusively. This is an extended form of hybrid cloud computing, called multi-cloud or multi-source solutions, also described by Bohli et al. [27] and Li et al. [28]. Actually, the classification if its a hybrid or a public cloud solution depends mainly on the point of view. As mentioned, most cloud solutions in practice are hybrid (multi-) cloud ones.

C. Hybrid key management system

Similar to the identity management system classification described by Hussain [29], we can separate key management systems in user centric and federated systems. While a user centric key management results in a high overhead for the consumer, by keeping the keys in a secure way, this is outsourced in federated case to a third party or the provider of the service. Just like the hybrid deployment model, we can use a hybrid key management model. The user keeps a master key as the root of an encrypted key tree, like described by Zarandioon et al. [30], while the tree is managed by the SME. As a result the SME can not read out plaintext data without the consumer's permission. Figure 2 illustrates the basic idea. In fact, a similar concept is used by Apple's instant messenger service iMessage [31]. Despite some privacy issues described by QuarkLabs [32], in their opinion this cloud based instant messenger service could be considered as the most practical and secure real-time messaging system available. As a result, we think a hybrid key management system, combined with state-of-the-art cryptography can be considered as highly realistic for practical use in future work for our approach.

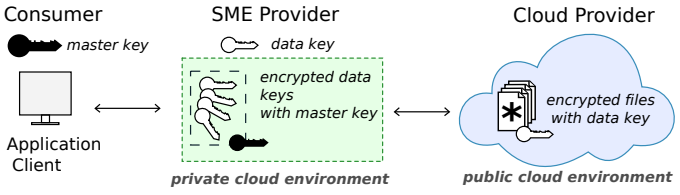


Figure 2. The hybrid key management system consists of a master key stored by the consumer and data keys stored by the SME provider. The data keys are encrypted by the consumer's master key.

IV. Scalable Encryption Layer

The scalable encryption layer is an additional tier between the application servers and the data storage server in the public cloud. Its location is illustrated by the green dashed box in Figure 1. As the figure shows the EL consists of two components, the encryption and the key management service. These services extend the proxy-like behavior of the CryptDB MySQL proxy developed by Popa et al. [33]. The basic idea of our EL is the encryption and decryption of data while transferring the data to the outsourced data storage (on-the-fly encryption). Therefore, less trust to the party to which the data has been outsourced is needed. To prevent the EL becomes kind of a bottle neck, it is located in a private cloud. This private cloud is controlled and managed by the SME provider and offers an efficient possibility for scaling.

Nevertheless, this approach is accompanied with some issues. First of all, virtual cloud instances should be as stateless and independent as possible. So, how and where to persist the keys need for encryption and decryption? Second, because of scaling and load balancing the following scenario is very common: Instance A encrypts some file, but the decryption request is forwarded to instance B. How to exchange encryption keys, if the instances should be independent? Third, the encryption layer should support the encryption of both structured and unstructured data. Which instance should decide which type of data is send to the EL and how it should be forwarded (in sense of load balancing). Fourth, despite the highly dynamic cloud environment there has to be some kind of a cloud access point, the application server can address their requests to. In this scenario this access point needs to know what instances exist, to be able to forward requests to them. Last but not least, all these problems and requirements should be solved in a way so that the existing system has to be adjusted as little as possible (low migration effort). Summing up, there are the following problems to solve:

1) Persistence and distribution of encryption keys.
2) Load Balancing, request forwarding with respect to the type of data.
3) Cloud instance controlling and addressing.
4) Low migration effort.

A. Technical setup and implementation of the protoype

The very first point to clarify is, which cloud management system is used for the private cloud. Important to mention is, this cloud management system only provides the cloud environment. That means, especially in the field of scaling, there are tools to easily start and stop virtual instances. However, in which way the implemented system reacts to upscaling (include for load balancing) or downscaling (exclude from load balancing) events, has to be managed by the system itself. Giving some thoughts, this fact is very clear, in an IaaS cloud environment the management system usually does not know what is inside these virtual instances and can therefore not react in any specific behavior. This is the main difference to PaaS solutions, where the management gets much more context information from source code, deployment rules and configuration files. In consideration of these facts a PaaS solution would be the favorable solution, but the effort to setup this solution is much too high for this prototype. So, we decided to use OpenNebula 4.4 [34] for our private cloud environment. Reasons for this decision are the open source environment, good possibility for own integrations and, last but not least, the fact of existing knowledge about OpenNebula. Our private cloud is powered by four physical hosts with 3 GHz Dual-Cores and 8 GB RAM. These machines consist of standard components and are connected via a common 100 MBit/s ethernet network to keep the hardware costs low.

After installation and setup the environment we had to cope with a very basic problem, to which we refer to as *image persistence dilemma*. Basically, there are two possibilities to setup an image in OpenNebula, from which the virtual instances are created: First kind are persistent images. As the name implies, these images save the adjustments the user is doing during runtime of the virtual machine. Besides this, another advantage of this *stateful* image is the rapid boot time. However, there are some crucial disadvantages. First of all, the access is exclusive, which means there is no possibility for scaling, based on persistent images. This makes perfect sense in consideration of constancy. Secondly, because of the direct usage of the image in the cloud-internal image repository (e.g., SAN/NAS as possibilities to provide access to a data storage in networks), which results in fast boot times, but the runtime performance is lower than in non-persistent images. This is because of the higher access time for the network storage, in contrast to the local disk, resulting in a higher CPU *wait time*. Especially, we observe this behavior by execution of write heavy disc access tasks, e.g., compiling the CryptDB MySQL proxy.

The second kind are non-persistent images. Again, as the name implies, all changes done while runtime inside this VM, are lost when shutting down this VM. In addition, the boot time for this kind of images are much longer, because the host needs a local copy on its physical hard drive. Trivially, a significant amount of time is necessary to transfer an image of 10-20 GB over the network. The advantages of non-persistent images are the better runtime performance and, even more important, the possibility to have more than one virtual instance of this image. One can argue, that long boot time only happens once, because the second instance of the host could copy the local image. However, there are two important points. First of all the possibility of a local copy depends on the image format. Some formats, like *qcow* [35], support copy on write. That means the local changes were stored in a different place, which opens up the possibility for other virtual instances to use the same local image. However, the hypervisor installed in the physical host, must be aware of this. Although we used the qcow format, the images were copied again from the image repository. At this point we see potential for future work. The second point is in consideration of the performance and reliability it would be better to run the other instances of the image on as many different (physical) hosts as possible.

How to solve this image persistence dilemma? Actually, the public IaaS cloud provider we use for data storage server provides a simple solution. There are only persistent images. So, if you want to scale up, you have to clone your images as often you want to scale up. This is inefficient in a lot of ways. First, it takes a lot of storage, resulting in high costs and inefficiency, because most of the time the images are not used. Second, there is an up-scaling limit in short-term by the number of cloned image. Third, because of the different images there are a lot of different states within the same virtual instances, making potential failure analysis extremely difficult.

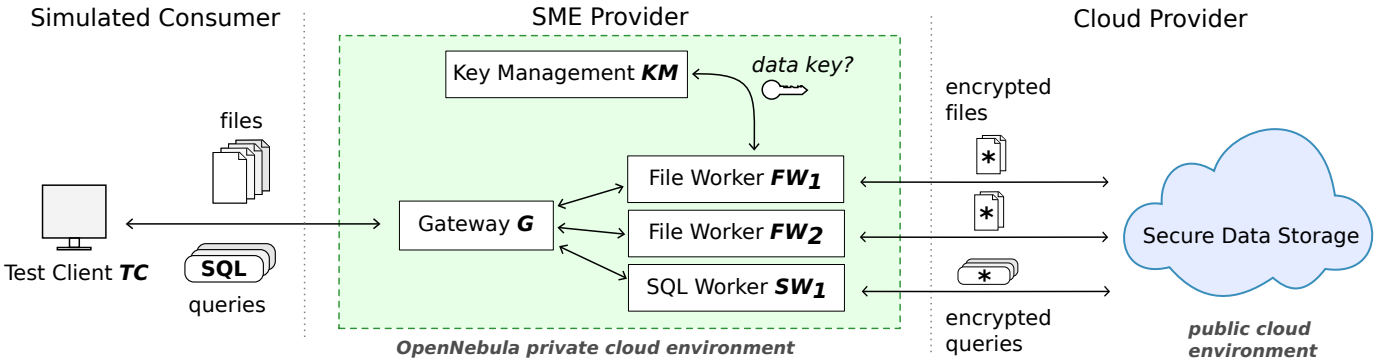


Figure 3. The implemented encryption server prototype with the Test Client TC to simulate consumer requests (files and SQL queries), our Encryption Layer (green dashed box) to encrypt/decrypt consumer requests and the data storage to persist the encrypted files/databases. The data flow is illustrated as well, showing only encrypted data leaves the private cloud environment.

However, in this prototype we have no need for scaling in the public cloud, but for productive systems this should be a crucial point while choosing a public cloud provider.
All in all, there is no perfect solution of the image persistence dilemma. The most common solution is the usage of a cluster file system, distributed over the hosts with high speed and high bandwidth internal network. This is open for future work. It has to be mentioned, that this optimizations toward a highly performant cloud environment is not suitable for most SME. As we can see, even public cloud providers lack in providing efficient and practical solutions. Therefore, the practical suitability of setting up a highly efficient and performant private cloud environment is not included in the practicability discussion of the EL.

1) Setup of the cloud system - virtual machines: Figure 3 shows an overview of the architecture of the implemented prototype, with the Test Client TC, the EL and the cloud storage. As mentioned, the EL consists of a key management and an encryption component. The focus of our work is setting up and implementing the encryption part as the core component and as crucial point for practicability discussions for SME SaaS provider. Therefore, the key management server KM is a basic MySQL database server for storing the encryption keys for file encryption and a basic NFS server for the SQL query encryption. Despite the fact this is not suitable for productive systems, it fulfills the persistence problem in a very efficient way and is therefore a good solution for our prototype implementation. Because the keys must be stored persistently and KM will not scale, it runs as a VM based on a persistent image.
As Figure 3 shows, besides KM the EL consists of a Gateway G, File Workers FW_i and a SQL Worker SW_1. With the decision of gateway solution, we address the problems of load balancing, forwarding, cloud instance controlling and addressing, and the low migration effort, by the risk of a single point of failure. In addition, the workers become more stateless and independent from application servers. We think for the prototype implementation this tradeoff is suitable. The gateway acts as a load balancer based on a JBoss AS 7 cluster, which also solves possible problems with internal communication between highly dynamic VMs. It also acts as a MySQL proxy, forwarding all database requests to the SQL Worker SW_1. As a result, the gateway behave for the application server (Figure 1) as a file server and database. This fact reduces migration effort significantly. Because the gateway does not need to store any data permanently, the image is non-persistent. Actually, there is the possibility to scale up the gateway. However, our hardware setup was not suitable to test such large scenario.
Another non-persistent image is used by the File Worker FW_1. These workers perform the encryption and decryption task and act as stateless and scalable nodes in a JBoss Cluster. As Figure 3 shows these workers request encryption keys from the key management server and upload files into the public cloud.
We ran in the problem how to deploy an application archive in the JBoss nodes. Because it is very inefficient to shut down FW_1, set the image to persistent state, start it again, deploy the archive, shut down, and switch back to non-persistent state to make it scalable again. After that we can start up the service again. Our solution will instead of pushing the new archive version into the virtual machine, pull it while starting up the VM. This is possible by the script shown in listing 1, which is processed while booting up the VM.

Listing 1. Application archive update/retrieval script

```
curl -Lkv -o /tmp/servlet.war -u <name>:<pw> '<URL>'
rm <JBOSS_DEPLOY_DIR>/servelet*
mv /tmp/servlet.war <JBOSS_DEPLOY_DIR>servlet.war
```

The first command fetches the servlet archive from the given $< URL >$. In our case, we get the latest version of the application archive stored on our internal repository management system for software artifacts (Sonartype Nexus [36]). Second and third command remove a possible old version and move the new application archive into the deployment directory of the JBoss application server. Actually, this naturally enables us to update an application while runtime, just by restarting a VM (or by triggering the script manually at VM runtime, therefore, the JBoss hot-deploy mechanism has to be active).
The SQL Worker SW_1 as the fourth image is also non-persistent. Initially, we plan to scale the SQL worker as well. However, it was not possible to outsource the key management of the CryptDB-enabled MySQL proxy deployed in SW_1 to KM with reasonable effort. Therefore, we outsource the database files of the internal database of the CrytptDB via NFS to KM. Trivially, if more than one SQL worker would (over)write these database files the consistency could not be ensured. As result we could not scale the SQL Worker.
All VMs are part of an autoscaling service of the OpenNebula cloud environment called *OneFlow*[37]. This service allows to

bundle some VMs and to set up a set of rules for scaling the virtual machines. OneFlow enables to adjust the number of VMs at the startup and how much load a VM has to have to scale up or down. Starting up our service with these five VMs takes around 10 min.

2) Setup of the JBoss Cluster: Figure 4 shows the structure of the internal JBoss Cluster, as described by Marchioni [38]. This cluster is used for load balancing in a dynamic environment. The cluster consists of a controller, which is basically an extended HTTP server located in the Gateway VM G. Another part of the cluster are nodes, that are configured JBoss AS 7 application server. The JBoss cluster addresses problems 2) Load Balancing and 3) Cloud instance control mentioned at the beginning of Section IV. So, this cluster is perfectly suited for the management of dynamically added and removed virtual machines. However, there is a limitation. As the HTTP server based controller suggests, the cluster only supports HTTP request. Therefore, it can not be used to load balance SQL query requests for the SQL Worker SW_1 and works only for the File Workers FW_i. Though, in consideration of the fact that only FW_i are scalable, this is perfectly fine.

The JBoss cluster is based on mod_cluster 1.2.6 [39]. The setup of the JBoss cluster consists of two main aspects: setting up the controller and setting up the nodes. As mentioned, the controller is located in G. To make the standard apache 2 HTTP server work as a JBoss cluster controller, an extension by the mod_cluster module is necessary. For configuration details please see [40]. To setup a standard JBoss AS7 as a cluster node, it is necessary to enable the mod_cluster module and configure it appropriately. Especially the multicast ports have to be the same as configured in the controller in G. The cluster communication is illustrated in Figure 4 and works like following. The controller in G sends out a multicast, containing the controller address information. As the nodes receive this multicasts they answer to the appropriate controller address, containing node information like deployed application archives and node load-metrics. Finally, the controller receives these node answers and can take them into account for load balancing. This principle is perfectly suited for dynamic cloud environments, as nodes can start and stop at any time.

For load balancing, mod_cluster provides a lot of load metrics. Our metrics are shown in the listing 2. For detailed meaning of the parameters please see [41].

Listing 2. Load balancing configuration

```
<dynamic-load-provider history="10" decay="2">
 <load-metric type="cpu" weight="2" capacity="1"/>
 <load-metric type="mem" weight="4" capacity="512"/>
 <load-metric type="ST" weight="1" capacity="512"/>
 <load-metric type="RT" weight="1" capacity="512"/>
</dynamic-load-provider>
```

G uses this metrics to calculate the busyness b of the node (weighted average), after the equation,

$$b = \frac{2 * cpu + 4 * mem + ST * RT}{8} \quad (1)$$

in which ST stands for *send-traffic* and RT stands for *received-traffic*. It has to be pointed out that the cpu metric in our virtual environment does not behave in the intended manner. We cloud not evaluate the exact reason, but numerous tests show that the configuration above leads to better results than one without the cpu metric.

In order to establish a high data security, we chose an at least 256 bit standard encryption method. Therefore, we have to use another Java security provider, because the standard security provider of Java only supports up to 128 bit encryption key length. So, we decided to use the security provider of Bouncycastle [42]. To use it in the server components in the nodes, we have to extend the JBoss AS7 nodes. First of all we replace the standard Java policy files with those from Bouncycastle. By default it is located under a path like *JAVA_HOME_DIR\jdk1.7.x_xx\jre\lib\security*. Secondly, a new module has to be added in the JBoss AS7 nodes. Therefore, the creation of a folder like *JBOSS_HOME_DIR/modules/org/bouncycastle/main* is necessary. The Bouncycastle library files have to be moved in this folder. The next step is the creation of the *module.xml* as shown in listing 3.

Listing 3. Bouncycastle module

```
<module xmlns="urn:jboss:module:1.1"
name="org.bouncycastle">
 <resources>
 <resource-root path="bcprov-jdk15on-150.jar"/>
 </resources>
 <dependencies>
 <module name="javax.api" slot="main"export="true"/>
 </dependencies>
</module>
```

The last step is to integrate this module, as listing 4 shows, in the JBoss AS7 server configuration file. One example could be *JBOSS_HOME_DIR/standalone/configuration/standalone.xml*.

Listing 4. JBoss AS7 server configuration

```
<global-modules>
      <module name="org.bouncycastle" slot="main"/>
</global-modules>
```

Summing up, to set up the JBoss Cluster as appropriate environment for the EL consists of three steps. The basic setup with controller in the gateway G and nodes in FW_i. Second, the configuration of the load balancing metrics and third, the extensions of the nodes to support 265 bit encryption standard methods.

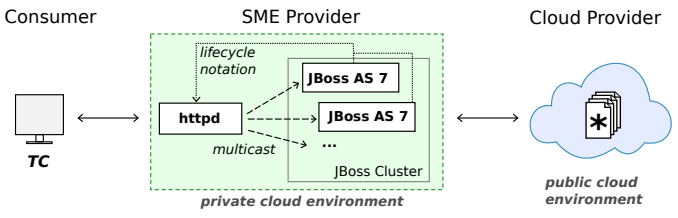
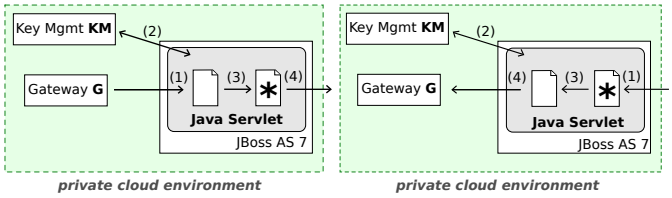
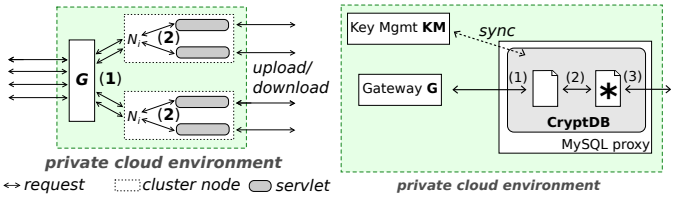


Figure 4. The internal JBoss Cluster inside the EL (green dashed box) for load balancing purposes. The figure shows the mod cluster enabled HTTP daemon (httpd) and the JBoss 7 application server nodes (JBoss AS 7), as well as the internal cluster communication to manage these JBoss Cluster nodes.



(a) file encryption (b) file decryption

Figure 5. Work and data flow inside the File Worker in the EL (green dashed box). Figures show encryption and decryption, respectively. The star symbolizes an encrypted file.



(a) Two Stage Scaling (b) CryptDB proxy

Figure 6. Two stage scaling while file processing inside the EL (green dashed box) and the work/data flow of the SQL query encryption.

3) Details of the Server Components: The server components of the encryption layer consist of a Java Servlet for file encryption and a MySQL proxy, based on the work of Popa et al. [33]. Figure 5(a) and Figure 5(b) show the procedure of file encryption and decryption, inside the JBoss AS7 Cluster nodes.

File encryption consists of four steps. First, receiving the HTTP request from the HTTP server (httpd) in gateway G (1). Second, checking if the file exists or creating a new key by sending a request to the key management system (Key Mgmt) (2), via a JDBC connection. Third, encrypting the file with a 256 bit encryption method (3). Fourth, upload the encrypted file to a file server in public cloud (4), using sardine, a WebDav client for Java [43].

File decryption is roughly the same, vice versa. First, downloading the encrypted file from the public cloud (1). Second, getting the encryption key from key management (2). Third, decrypting the file (3). Fourth, sending the file to the gateway or client, respectively (4).

Since the file up and downloads work in a synchronous way, a node would not be able to process another request, while uploading or downloading a file. This problem is naturally solved by JBoss AS 7 servers, by allowing many instances of the servlet in parallel, using threads. Actually, this is the second stage of scaling in the Encryption Layer. The two stages are shown in Figure 6(a). First a coarse-grained scaling in G, based on load balancing and virtual machines and cluster nodes (1) and second a fine-grained, based on parallel threads inside the nodes N_i (2). As mentioned above, there is of course the possibility of scaling the whole system, by having a redundant gateway, cluster and so on. This scenario is not considered, because of limitations of the available hardware for testing the prototype.

The servlet was developed in consideration of logging results and benchmarking different encryption methods. Table II shows test results of different encryption methods. The results show that AES (256Bit) works most efficient, considering encryption and decryption times. As a result of this and because of the fact AES is widely accepted as a secure encryption method, all following tests use AES (256Bit) for encryption. As mentioned, for detailed performance tests different times were logged. Because the File Workers FW_i nodes were not persistent, the logged times had to be sent to a logging service or fetched by one. Actually, we did not implement a separated logging service. We decided, to add the logged times in the HTTP header of the resulting HTTP response. Despite the small overhead this was very efficient for the prototype. Because of this approach all relevant performance measurement time bunched in the Test Client TC, the analysis become more easier. Logged data is:

1) start and communication times with key management
2) encryption method, and times spent for encryption and decryption
3) start and communication times with public cloud (upload, download, delete)

The second part of the server components is the integration of the MySQL proxy based CryptDB approach. The setup is described in *doc* folder in sources of the git repository of CryptDB [44] and consists basically of setting up some environment parameter and compiling an extended MySQL proxy. Our problem was to outsource the internal key management system of the CryptDB out of the SQL worker. Since the CryptDB approach uses an InnoDB database for internal consistency and backups, our attempt to forward these internal request to a remote database did not succeed. Therefore, we outsourced the database files in a very pragmatic way via NFS, leaving an improved solution open for future work. Another problem is scaling with databases. As far as our search results showed up, there is no comparable open source load balancer like mod_cluster available for databases. Actually, there are solutions like pgpool [45] and plproxy [46]. However, these tools are limited to a fixed pool of instances. Therefore, it is not possible to dynamically add some additional virtual machine or stop one VM. It is questionable if it really make sense to add additional database instances on demand, because principle like master-slave and database sharding are not suited for highly dynamic or short timed setups, which makes perfect sense for a persistence layer. Hence, existing solutions like Relational Cloud [47] go another way.

4) Details of the Test Client Implementation: The implementation of the Test Client TC is completely done in Java. TC has to fulfill three main tasks. First of all, the possibility to send HTTP file and SQL query requests to the EL. Second, the setup of different flexible test scenarios. And third, to measure and log the performance of the EL to analyze the test results. The first task is easily done by basic HTTP GET, POST and

TABLE II. UPLOAD, DOWNLOAD, ENCRYPTION, AND DECRYPTION AVERAGE TIMES $\bar{t}$ IN MILLISECONDS (FILE SIZE 1MB)

encryption method	$\bar{t}_{up}$[ms]	$\bar{t}_{enc}$[ms]	$\bar{t}_{down}$[ms]	$\bar{t}_{dec}$[ms]
AES (265 bit)	1040.9	39.9	1116.4	51.5
DESede (168 bit)	1165.9	167.2	1239.4	135.8
Serpent (256 bit)	1180.9	57.2	1138.2	57.9
Twofish (256 bit)	1195.9	50.6	1160.4	50.5
CAST6 (256 bit)	1300.9	53.3	1037.6	40.1

DELETE requests and a JDBC connection. These protocols are also used by the application server. For a direct communication with the WebDav file server in the public cloud again the sardine client is used. Actually, the CryptDB approach supports JDBC only in a very limited way. Nevertheless, it is enough for our EL prototype, to enable basic SQL statements for our tests. For the second task, we implement a highly configurable and effective possibility to create different load scenarios. Therefore, we can simulate autonomous clients with an individual behavior in parallel.

Figure 7 shows the basic parameters for configuration of the simulated clients. The figure illustrates a blue load line separated in *slack* and *spike* sectors. A slack sector represents a normal, average number of requests which cause a basic load to the EL. A spike sector represents a load peak, with a lot of requests in a short time span to cause some heavy load. In addition, Figure 7 shows a spike sector is always surrounded by two slack sectors. There can be an arbitrary number s of spikes, normally we set $1 \leq s \leq 3$ to finish the test scenarios in a reasonable time. As shown, there are a number of upload/download/delete blocks, called UDD. These blocks symbolize the procedure per file, which is at first uploaded to public cloud (through EL), after that download the file and check for identity. If successful the file is deleted in the cloud and UDD has processed successfully. This order is fixed, as one can not download or delete the file, until it is fully uploaded. The number n of UDD_n is configurable to $1 \leq n \leq 12$. In addition, the figure shows delays $D_{slack}(C_j, n_{slack}) = \Delta T(UDD_n, UDD_{n+1})$ and $D_{spike}(C_j, n_{spike}) = \Delta T(UDD_n, UDD_{n+1})$. Were $\Delta T(x, y)$ represents the timespan between the starting points of x and y, C_j is the client number j, and n_{slack}, n_{spike} are the numbers of UDD blocks in slack and spike, respectively. Normally, $D_{slack} \gg D_{spike}$, to simulate longer basic load periods and short peak load periods. The UDD blocks are independent, which means that up to n blocks can be processed in parallel. Last but not least, there is an option to choose the file size $F(C_j)$ between 1MB(F_1), 10MB(F_{10}) and 100MB(F_{100}). This file size is than fixed for this client C_j. Summing up, the configuration protocol for C_j works like processing the following protocol:

1) How many spikes s do you want?
2) How many UUD blocks should be sent in slack sector? (set n_{slack})
3) What is the (expected) delay? (D_{slack})
4) Choose a file size between F_1, F_{10} and F_{100}...
5) *// Repeat step 2 - 4 for spike configuration, then go to step 6*
6) Do you want to add another client C_{j+1}?

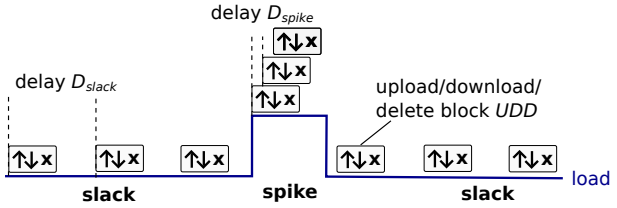


Figure 7. Client basic configuration for test scenarios with basic UDD blocks. One block consists of upload (↑), download(↓) and delete (x) of a file.

As question 6 suggests, there is the possibility to add another client. Basically, there is no limitation of the number of client we can set up. In case of adding another client, there will be a question how long the delay ΔT_{C_j} between the start of client simulation and start of C_j should be (simplification of $\Delta T(0, C_j)$). Slightly different is the setup of the SQL query client. In this case, we decided to implement a basic loop, consisting of a chosen number $N_{DML}(C_j)$ of data manipulations with INSERT, and DELETE statements, and a different number $N_{DQL}(C_j) \leq N_{DML}(C_j)$ of data queries with SELECT statements.

The third task, is done by logging the client configuration, the track of the configured protocol and by reading out the node-logs from the HTTP response headers. In addition, TC logs the status of the virtual machines of the private cloud environment. Therefore, the API of OpenNebula is used to log CPU and memory usage and in/outgoing network traffic for each VM. Because the OpenNebula System updates these values once every 20 seconds, the client fetches these VM performance measurements in the same period of time. Consequently, the logged data is:

1) start time of UDD blocks (+ details) and clients
2) VM CPU and memory usage, and in/outgoing network traffic
3) conglomeration of cluster node logs

The last important function TC provides is to run a comparison test. In this test setup the configured, simulated clients not only send their requests to G, but send them directly to the public cloud as well. This offers a good possibility to measure the overhead of the Encryption Layer.

5) Setup of the Public Cloud Servers: An Apache 2 HTTP server [48] and WebDav [49] is used to provide a file server in the public cloud. The file server is hosted by an European IaaS provider. For storing this data a common MySQL database extended by CryptDB user defined functions is also hosted by the IaaS provider. The extension is done by adding the CryptDB library *edb.so* to the plugins of the MySQL database and installing the NTL Library [50]. Both, virtual server use minimal resources of 1 GHz and 1 GB RAM.

Table III shows the summary of the setup of the Encryption Layer. We can see that the VMs have a decent usage of resources. Even if our hardware consists of standard components, this setup works very well for our EL prototype. The work of Toraldo [51] confirms that our setup in the private cloud is realistic.

B. Testing the prototype

The tests of the EL prototype are separated into load balancing and overhead test. A third scenario, testing dynamic scaling is not possible out of two reasons. Number one is discussed above, we called it image persistence dilemma. In our prototype setup it takes up to 5 min for a new VM to be ready to answer requests, including the time to transfer the 10 GB image and to boot up the VM. The second, more important point is that our available network bandwidth limits the load we can create in the Encryption Layer, making it unnecessary to improve our local image repository storage solution. This point is argued in the discussion section.

TABLE III. OVERVIEW OVER ENCRYPTION LAYER SETUP

Domain	VM name	vCPU [GHz]	RAM [GByte]	installed software
private Cloud	key management	0.50	0.25	MySQL database
	gateway	0.40	0.50	Apache 2, mod_cluster, MySQL proxy
	file worker	0.50	2.00	JBoss, Bouncycastle, Java servlet
	SQL worker	1.50	1.00	MySQL proxy, CryptDB
public cloud	file server	1.00	1.00	Apache 2, WebDAV
	SQL server	1.00	1.00	MySQL database, CryptDB MySQL module

1) Load balancing test scenario: Our aim is to show that client requests are equally distributed over two or more VMs in a dynamic way. Therefore, the JBoss cluster is configured in the above mentioned way and 2 File Worker VMs run in the private cloud. Figure 8 illustrates the protocol of the simulated clients by TC, sticking to the introduced symbolics above in Figure 7.

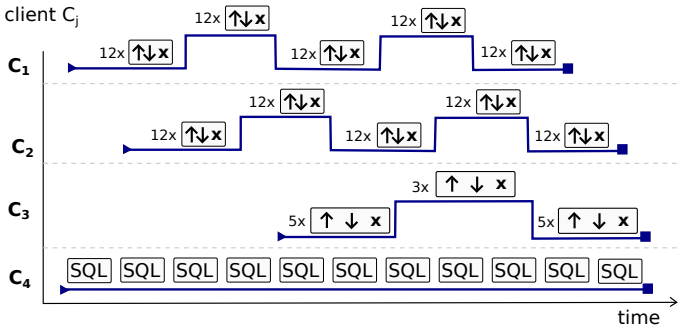


Figure 8. The illustration of the test protocol of the load balancing scenario. Four client are simulated to create load to the EL.

As shown, four clients are simulated, with the following parameters.

- $s_{C_{1,2}} = 2$, $s_{C_3} = 1$
- $\Delta T_{C_{1,4}} = 0$, $\Delta T_{C_2} = 20$ s, $\Delta T_{C_3} = 40$ s
- $D_{slack}(C_{1,2}, 12) = 2000$ ms
- $D_{spike}(C_{1,2}, 12) = 10$ ms
- $D_{slack}(C_3, 5) = 5000$ ms
- $D_{spike}(C_{1,2}, 3) = 10$ ms
- $F(C_{1,2}) = F_1$, $F(C_3) = F_{10}$
- $N_{DML}(C_4) = 15$, $N_{DQL}(C_4) = 10$

To complete the test protocol, the simulated clients take 10 min and 23 s.

2) Overhead test scenario: Our aim is to estimate the overhead of the EL. Therefore, a long running test has been set up. Figure 9 illustrates the protocol of the simulated client by TC.

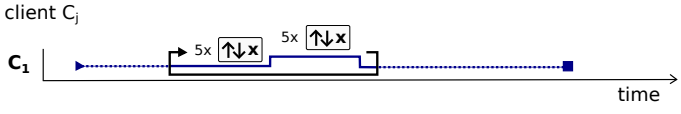


Figure 9. The test protocol for the long running overhead test scenario.

As shown, one clients is simulated, with the following parameters.

- $\Delta T_{C_1} = 0$,
- $D_{slack}(C_1, 5) = 20$ s, $D_{spike}(C_1, 5) = 20$ s
- $F(C_1) = F_1$

As Figure 9 and parameters show, this is a special setup with no peak load. In this scenario the spike symbolize the usage of our EL and the slack stands for a direct public cloud communication, without any encryption. Therefore, we can compare the results to get an estimation of the overhead. The test protocol is repeated until the user cancels it, in this case the test runs over 17 h.

Apart from these tests scenarios, we perform a lot of other tests to optimize parameters, like those of JBoss load balancing metrics, encryption methods and internal HTTP requests. We also run tests for SQL query processing comparison, by communicating directly with the database server in the public cloud. Despite of the given overhead values of CryptDB from Popa et al. [33], this enables us to estimate the overhead for SQL query encryption in our EL. This also allows to compare file encryption overheads with those from SQL query encryption.

C. Test results from the prototype

Like the test scenarios, the test results are separated in load balancing and overhead test results. The shown results are based on the data logged by TC, received from the JBoss cluster and the OpenNebula Management System. Figure 3 shows an abstract overview of the test setup. The detailed test scenarios are described in the section above.

1) Load Balancing Test Results: With the load balancing tests, we want to show how good the EL prototype can handle different load situations. The configured load balancing metric for the JBoss Cluster works very well. As mentioned, the metrics configured in mod_cluster config in JBoss nodes, combine CPU load, system memory usage and amount of outgoing/incoming requests traffic. Figure 10 shows the logged VM workloads of G and F_1, F_2 in 20 s intervals. The figure also show the number of client requests sent by TC in the specific interval. Please note that some client requests, like DELETE, do not cause much load. This is the reason, why the interval at 540 s has a low CPU load compared to the number of client requests. However, the timespan between 340 and 420 s is remarkable. A lot of upload and download requests were sent in this timespan. The gateway recognizes the high load of FW_2 sending client requests to FW_1. At time intervals of 380 s, it is inverse. In Figure 8 this is illustrated at the point when

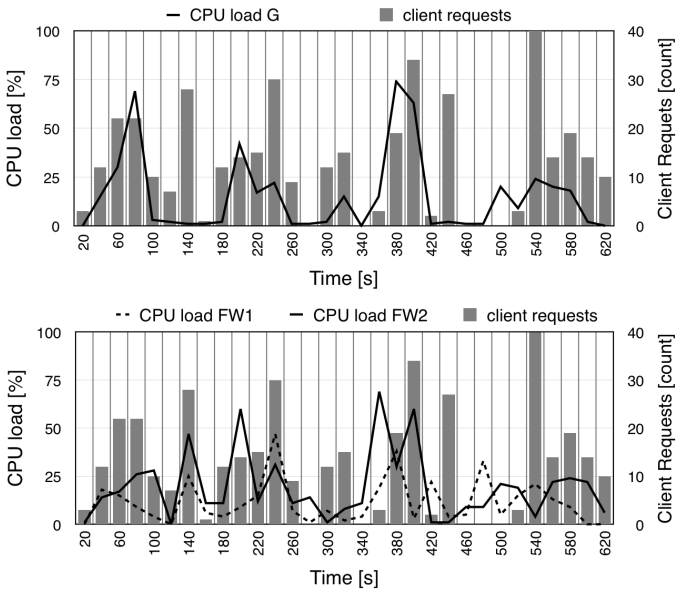
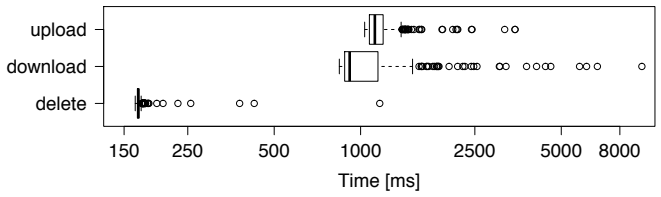
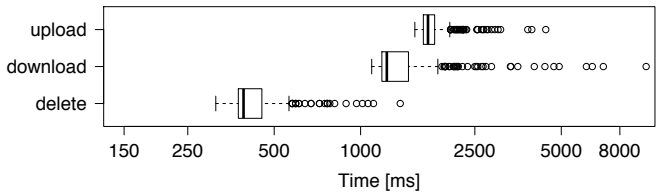


Figure 10. CPU load of G, FW_1, FW_2 VMs depending on the number of client requests.



(a) Direct cloud communication. Median times: Upload 1121 ms, Download 916 ms, Delete 162 ms



(b) Communication through Encryption Layer. Medians times: Upload 1717 ms, Download 1234 ms, Delete 391 ms

Figure 12. Logarithmic scaled response times for processing 515 uploads, 515 downloads and 515 delete file requests with and without the Encryption Layer. Whiskers maximums are 1.5 IQR

all clients $C_{1,2,3}$ have their spike loads at the same time. It has to be pointed out that Figure 8 is only a sketch, since clients C_i are completely independent and influenced by network latency and bandwidth. Figure 11 shows the response times for processing the SQL queries sent by TC though simulated client C_4 as illustrated in Figure 8.

2) Overhead Test Results: For the overhead test our aim is to get an estimation of a lower boundary of additional effort we have to accept, if we want to use on-the-fly encryption methods like an EL. Therefore, we ran some long term tests, to compare direct cloud communication with that through our EL. We uploaded, downloaded and deleted files in this test over 17 h. Figure 12(a) and 12(b) show the result box plots without and with the EL, respectively. Please note that the x-axis has a logarithmic scale. In Figure 12(a) we can see a very compact box plot for deleting files. This means the median of 168 ms for the deletion a file in the cloud storage is, except of very view outliers, nearly constant. Similar results can be seen in cases of upload and download the files. However, since these processes take much longer time, they are more influenced by a fluctuating network latency and bandwidth. Interestingly, the download is influenced much more. Figure 12(b) illustrates the box plots while using our EL. We can see the expected higher communication times. In addition, we can see that the data distribution in the box plot looks analogue to those in Figure 12(a). As we expected the delete time increases most. This is explainable because of the additional roundtrip to the key management system to delete the encryption key.

In another test we measured the response times of the database in cloud storage without our EL communication. Figure 13 illustrates the results. Please note that Figure 11 and Figure 13 have a different scale. Otherwise, the data is no longer readable, because of the high difference. Figure 11 shows results of the 18 rounds the SQL client C_4 executed its protocol. Remarkable in Figure 13 is that response times do have very few outliers. This is explainable in the short communication time and the short overall test time of around 6 min. In this short timespans the network latency fluctuations do not influence the results. Hence, the many outliers in Figure 11 do not result of a fluctuating network latency, since the duration of the test was only around 10 min. With regard to the logarithmic scale, a response times of ten to twenty seconds for a very basic query are unacceptable for practical use. Our best guess is the CryptDB proxy doing some internal recovery and key management operations. However, as mentioned we bind the relevant folder via NFS to our key management KM, some file operations via the internal network should not last that long. Figure 14 shows the percentage of times for processing a request through the EL. The figure splits up again in upload,

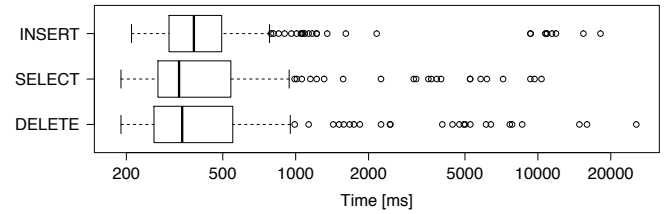


Figure 11. Logarithmic scaled response times for processing 234 INSERT, 180 SELECT and 234 DELETE queries through the Encryption Layer. Whiskers maximums are 1.5 IQR.

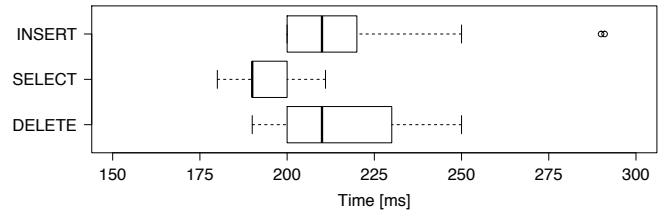


Figure 13. Response times for processing 310 INSERT, 340 SELECT and 310 DELETE queries direct to cloud. Whiskers maximums are 1.5 IQR.

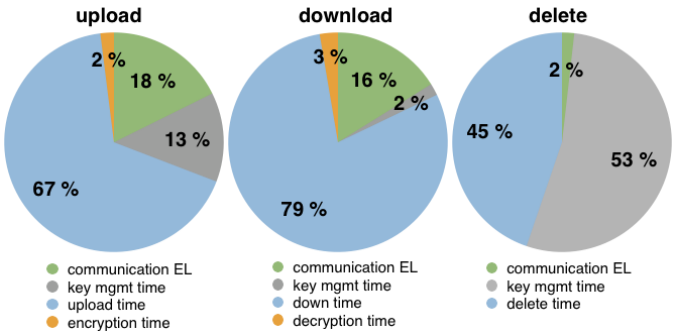


Figure 14. Percentage of times to encrypt and upload, decrypt and download, and delete a file, respectively.

download and delete parts. In cases of up- and download the communication time includes times spend for up- and download the file from TC to EL, therefore, the percentage is much higher than in case of deletion. More important is the fact that only two/three percent is used for encryption/decryption, the rest of time the File Workers are, roughly speaking, waiting to complete communications. Especially the communication overhead while deleting a file is significantly high, taking more than 50% of overall time.

The overheads for file encryption and query encryption can be seen in Figures 15 and 16. In fact, the figures illustrate the results of Figure 12, and Figures 11 and 13, respectively. In numbers, the overhead to upload and download a file, displayed in Figure 15 is around 53% and 34%, respectively. The overhead to delete a file is with around 132% very high. The difference can be explained by the required effort to store/fetch/delete the encryption keys and is also illustrated in Figure 14. As displayed in Figure 16 the overheads for query encryption are not as divers as the file encryption overheads. Nevertheless, their absolute values are higher, 81% for INSERT, 74% for SELECT and 62% for DELETE statements. It has to be pointed out, both Figures 15 and 16 display median, not average, times.

V. Discussion

We lead the discussion under the aspects of performance and overhead of the EL, and development and operation effort for SME provider. As mentioned, we used median times in our statistics. The reason is to estimate the lower boundary of additional effort for a scalable on-the-fly encryption system. Therefore, median times give a much better statement, because network or computation fluctuations do not influence that much. This is especially interesting to compare the SQL query encryption. As we figured out in our previous work [1], the current development status is highly prototypic and not usable for productive systems. Nevertheless, for an estimation we do need comparable values. If we assume a weighted average of 30% uploads/INSERTs, 60% downloads/SELECTs and 10% deletes, we get the following lower boundaries of average overhead $\overline{\Omega}$.

$$\overline{\Omega}_{fileEnc} = \frac{30 * 53 + 60 * 34 + 10 * 132}{100} = 49.5 \quad (2)$$

$$\overline{\Omega}_{queryEnc} = \frac{30 * 81 + 60 * 74 + 10 * 62}{100} = 74.9 \quad (3)$$

So, we got $\overline{\Omega}_{fileEnc}$ of around 50% and $\overline{\Omega}_{queryEnc}$ of around 75% for overall processing. We can see, that database encryption is more complicated than file encryption. This can be concluded from the fact that database encryption is not only a matter of data-at-rest encryption, but computation under encryption as well. However, the support of databases is limited in a way not all queries can be supported; details can be seen in Tu et al. [52].

As we can see in Figure 14 the main overheads results from communication with KM and EL. The time used for encryption and decryption is with two/tree percent very low. So, on the one hand, we agree with Huang and Xiaojiang [53], that protecting data with cryptography methods cause unavoidable overheads. However, on the other hand, we disagree with the statement, that it introduces heavy computational overhead. As we can see from Figure 14 the computational overhead is nearly negligible, most overhead results from network communication. So, network communication is the most influencing factor for the performance in our approach. Because the limiting network bandwidth of the internet access was the main reason we could not run scalability tests, since the computational effort for encryption/decryption was so low.

VI. Related work

Our approach applies typical security concepts in the field of cloud computing using tier, logic and data partitioning described by Bohli et al. [27]. In contrast to this study, we do not spread our tiers to various, non-collaborating cloud providers. We spread our solution over a private and public

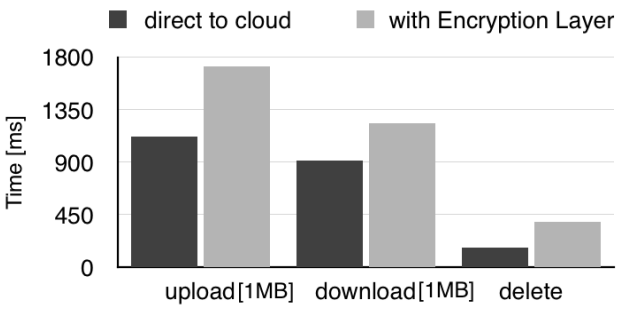


Figure 15. Diagram of median times to upload, download, and delete files with and without encryption

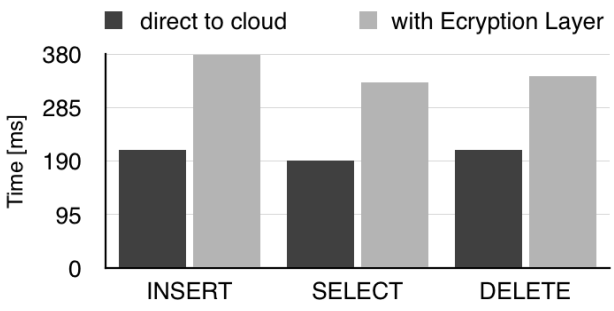


Figure 16. Diagram of median times to INSERT, SELECT, and DELETE queries with and without encryption

cloud, performing all critical tasks in the private cloud, like encryption and key management and outsource only the data tier in public cloud. This makes it unnecessary to label tasks or data as critical in a manual or semi-automatic manner, like described by Zhang et al. [21] and Oktay et al. [23]. Equally, Ray and Ganguly [22] present a data privacy model for cloud computing, in which sensitive and non-sensitive data is maintained separately. Zhou et al. [54] describe an evolved approach, using privacy-aware data retrieval, addressing problems of anonymization like quasi identifiers. Wang et al. [16] give a short overview over hybrid cloud security. Their *Single Encryption* scenario is in that way similar to ours that data is encrypted in private cloud before transferring in public cloud. However, their focus lay on authentication model for inter (multi-) cloud communication. A data management method, via a data portfolio, for company data in a hybrid cloud configuration, is discussed by Tanimoto et al. [24]. Li et al. [28] describe a framework for SME to orchestrate cloud services in multi-cloud environments. Their work tries to integrate applications of the internal information system of SME with a public e-business platform.

VII. Future work

An appropriate prototype of the discussed hybrid key management system is one of the very next steps in future work, in order to have a fully functional hybrid cloud environment. In addition, we could not run scaling tests because of limited hardware resources, we leave this open for future work. Our test results confirm the mentioned fact by Popa et al. [33] that the implementation of CryptDB is highly prototypical. As the own implementations of Google and SAP show (for details see [55]) more development effort, e.g., towards full support of JDBC - is necessary. Figure 14 points out that the encryption workload of file worker is not high. This leaves space for additional functionalities like file compressing, for faster up- and downloads, or file indexing for possible searches over the encrypted files. The latter is mentioned in [56]. Also, the integration of a secure identity and key management system, e.g., Kerberos [57], is required to provide a SaaS solution with focus on the customers privacy. Moreover, the tests show that implementations of failure and backup routines are absolutely necessary. Despite the different focus in this first implementation, we want to point out that security is not only about protecting data from unauthorized access or viewing, but also issues of auditing, data-integrity, and reliability should be concerned too.

VIII. Conclusion

In this this paper the introduction of an additional architecture layer, called Encryption Layer (EL) is described. With this layer SME can address privacy and security issues through the usage of encryption methods. SME have the option to basically employ three cloud models (private, public and hybrid) as their solution, which were discussed and compared in Section II. The private cloud model is the most inflexible and cost intensive option. Probably being an option for large companies owning much hardware resources, it is not suitable for SME. The public cloud model is the preferred choice for SME if the application has no particular high privacy or security requirements. Our work, which is focused on a hybrid cloud concept, offers a compromise between security/privacy, efficiency and costs. Therefore, we describe our developed prototype of the EL in detail. The prototype includes scalable encryption server for files and queries, and a basic key management system inside a private cloud environment based on OpenNebula 4.4. In addition, we set up file and database server in a public cloud environment. Problems as the so called image persistence dilemma were discussed and appropriate solutions are suggested. Moreover, we proposed a hybrid key management system, which can be used to distribute key management task between consumer and SME in order to achieve a higher privacy for service customers. Our test results showed that load balancing inside the private cloud, using a JBoss Cluster works well. Besides, we showed the main percentage of the overheads of 50% - 75% is a matter of communication, and not of heavy computation because of encryption and decryption. Therefore, network communication is the most influencing factor for performance. To be applicable in productive systems, improvements of performance and more research are necessary. Nevertheless, we show that SME can establish solutions like our EL to solve security and privacy issues with a reasonable amount of effort. Our work provides a good basic solution for SME to get their first experiences in the cloud computing business.

Acknowledgment

The published work has been financially supported by the European Social Fund (ESF) and the EU. We would like to thank the anonymous reviewers for helpful comments.

References

[1] P. Reinhold, W. Benn, B. Krause, F. Goetz, and D. Labudde, “Hybrid cloud architecture for software-as-a-service provider to achieve higher privacy and decrease security concerns about cloud computing,” in CLOUD COMPUTING 2014, The Fifth International Conference on Cloud Computing, GRIDs, and Virtualization, 2014, pp. 94–99.

[2] Instagram. Website. [Online]. Available: http://instagram.com [retrieved: November, 2014]

[3] Dropbox. Website. [Online]. Available: https://www.dropbox.com [retrieved: November, 2014]

[4] F. Gens, “It cloud services user survey, pt. 2: Top benefits & challenges,” IDC eXchange, 2008.

[5] ——, “New idc it cloud services survey: top benefits and challenges,” IDC exchange, 2009.

[6] BITOM and KMPG, “Cloud-monitor 2013 cloud-computing in deutschland – status quo und perspektiven,” KMPG Study, February 2013.

[7] CrispResearch. Study platform-as-a-service: German sme market survey. [Online]. Available: http://www.business-cloud.de/wp-content/uploads/2014/07/STUDIE-Platform-as-a-Service01.pdf [retrieved: November, 2014]

[8] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, and D. Leaf, “Nist cloud computing reference architecture,” Tech. Rep., 2011.

[9] P. Mell and T. Grance, “The nist defintion of cloud computing,” National Institute of Standards and Technology, Tech. Rep., 2011.

[10] E. Aguiar, Y. Zhang, and M. Blanton, “An overview of issues and recent developments in cloud computing and storage security,” in High Performance Cloud Auditing and Applications. Springer, 2014, pp. 3–33.

[11] S. Jie, J. Yao, and C. Wu, “Cloud computing and its key techniques,” in Electronic and Mechanical Engineering and Information Technology (EMEIT), 2011 International Conference on, vol. 1. IEEE, 2011, pp. 320–324.

[12] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," Journal of Network and Computer Applications, vol. 34, no. 1, 2011, pp. 1–11.

[13] J. Rhoton and R. Haukioja, Cloud Computing Architected: Solution Design Handbook, 2013th ed. Recursive Press, 2013.

[14] D. A. Fernandes, L. F. Soares, J. V. Gomes, M. M. Freire, and P. R. Inácio, "Security issues in cloud environments: a survey," International Journal of Information Security, vol. 13, no. 2, 2014, pp. 113–170.

[15] S. Van Hoecke, T. Waterbley, J. Devos, T. Deneut, and J. De Gelas, "Efficient management of hybrid clouds," in CLOUD COMPUTING 2011, The Second International Conference on Cloud Computing, GRIDs, and Virtualization, 2011, pp. 167–172.

[16] J. K. Wang and X. Jia, "Data security and authentication in hybrid cloud computing model," in Global High Tech Congress on Electronics (GHTCE). IEEE, 2012, pp. 117–120.

[17] M. A. Rahaman. How secure is sap business bydesign for your business. [Online]. Available: http://scn.sap.com/docs/DOC-26472 [retrieved: November, 2014]

[18] C. Gentry, "A Fully Homomorphic Encryotion Scheme," Ph.D. dissertation, Stanford University, 2009.

[19] R. A. Popa, E. Stark, J. Helfer, S. Valdez, N. Zeldovich, M. F. Kaashoek, and H. Balakrishnan, "Building web applications on top of encrypted data using mylar," in USENIX Symposium of Networked Systems Design and Implementation, 2014.

[20] M. van Dijk, A. Juels, A. Oprea, R. L. Rivest, E. Stefanov, and N. Triandopoulos, "Hourglass schemes: how to prove that cloud files are encrypted," in Proceedings of the 2012 ACM conference on Computer and communications security. ACM, 2012, pp. 265–280.

[21] K. Zhang, X. Zhou, Y. Chen, and X. Wang, "Sedic : Privacy-Aware Data Intensive Computing on Hybrid Clouds Categories and Subject Descriptors," 2011, pp. 515–525.

[22] C. Ray and U. Ganguly, "An approach for data privacy in hybrid cloud environment," in Computer and Communication Technology (ICCCT), 2011 2nd International Conference on. IEEE, 2011, pp. 316–320.

[23] K. Y. Oktay, V. Khadilkar, B. Hore, M. Kantarcioglu, S. Mehrotra, and B. Thuraisingham, "Risk-aware workload distribution in hybrid clouds," in Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on. IEEE, 2012, pp. 229–236.

[24] S. Tanimoto, Y. Sakurada, Y. Seki, M. Iwashita, S. Matsui, H. Sato, and A. Kanai, "A study of data management in hybrid cloud configuration," in Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2013 14th ACIS International Conference on. IEEE, 2013, pp. 381–386.

[25] Amazon. Aws s3. [Online]. Available: http://aws.amazon.com/s3/ [retrieved: November, 2014]

[26] Microsoft. Azure sql database. [Online]. Available: http://azure.microsoft.com/en-us/services/sql-database/ [retrieved: November, 2014]

[27] J.-M. Bohli, N. Gruschka, M. Jensen, L. L. Iacono, and N. Marnau, "Security and privacy-enhancing multicloud architectures," Dependable and Secure Computing, IEEE Transactions on, vol. 10, no. 4, 2013, pp. 212–224.

[28] Q. Li, Z.-y. Wang, W.-h. Li, J. Li, C. Wang, and R.-y. Du, "Applications integration in a hybrid cloud computing environment: modelling and platform," Enterprise Information Systems, vol. 7, no. 3, 2013, pp. 237–271.

[29] M. Hussain, "The Design and Applications of a Privacy-Preserving Identity and Trust-Management System," Ph.D. dissertation, Queen's University (Kingston, Ont.), 2010.

[30] S. Zarandioon, D. D. Yao, and V. Ganapathy, "K2c: Cryptographic cloud storage with lazy revocation and anonymous access," in Security and Privacy in Communication Networks. Springer, 2012, pp. 59–76.

[31] Apple. ios security. [Online]. Available: http://images.apple.com/iphone/business/docs/iOS_Security_Feb14.pdf [retrieved: November, 2014]

[32] Quarkslab. imessage privacy. [Online]. Available: http://blog.quarkslab.com/static/resources/2013-10-17_imessage-privacy/slides/iMessage_privacy.pdf [retrieved: November, 2014]

[33] R. A. Popa, C. Redfield, N. Zeldovich, and H. Balakrishnan, "Cryptdb: protecting confidentiality with encrypted query processing," in Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles. ACM, 2011, pp. 85–100.

[34] OpenNebula. Website. [Online]. Available: http://docs.opennebula.org/4.4/release_notes/ [retrieved: November, 2014]

[35] QEMU. Qcow3. [Online]. Available: http://wiki.qemu.org/Features/Qcow3 [retrieved: November, 2014]

[36] Sonartype. Nexus. [Online]. Available: http://www.sonatype.org/nexus/ [retrieved: November, 2014]

[37] ONEFlow. Documentation. [Online]. Available: http://docs.opennebula.org/4.4/advanced_administration/application_flow_and_auto-scaling/appflow_configure.html [retrieved: November, 2014]

[38] F. Marchioni, JBoss AS 7 Configuration, Deployment and Administration. Packt Publishing Ltd, 2011.

[39] ModCluster. mod-cluster website. [Online]. Available: http://www.jboss.org/mod_cluster [retrieved: November, 2014]

[40] ——. Documentation. [Online]. Available: http://docs.jboss.org/mod_cluster/1.2.0/html/ [retrieved: November, 2014]

[41] Redhat. Mod cluster subsystem. [Online]. Available: https://access.redhat.com/documentation/en-US/JBoss_Enterprise_Application_Platform/6.1/html/Administration_and_Configuration_Guide/Configure_the_mod_cluster_Subsystem.html [retrieved: November, 2014]

[42] Bouncycastle. Website. [Online]. Available: http://www.bouncycastle.org/java.html [retrieved: November, 2014]

[43] G. R. Sardine. Sardine - an easy to use webdav client for java. [Online]. Available: https://github.com/lookfirst/sardine [retrieved: November, 2014]

[44] Git-Repository. cryptdb. [Online]. Available: git://g.csail.mit.edu/cryptdb [retrieved: November, 2014]

[45] Pgpool. Website. [Online]. Available: http://www.pgpool.net/mediawiki/index.php/Main_Page [retrieved: November, 2014]

[46] Plproxy. Website. [Online]. Available: http://plproxy.projects.pgfoundry.org/doc/tutorial.html [retrieved: November, 2014]

[47] C. Curino, E. P. C. Jones, R. A. Popa, N. Malviya, E. Wu, S. Madden, H. Balakrishnan, and N. Zeldovich, "Relational cloud: A database-as-a-service for the cloud," in 5th Biennial Conference on Innovative Data Systems Research, CIDR, 2011.

[48] Apache. Apache 2.2 website. [Online]. Available: http://httpd.apache.org/docs/2.2/en/ [retrieved: November, 2014]

[49] WebDAV. Website. [Online]. Available: http://www.webdav.org [retrieved: November, 2014]

[50] NTL-Library. Ntl: A library for doing number theory. [Online]. Available: http://www.shoup.net/ntl/ [retrieved: November, 2014]

[51] G. Toraldo, OpenNebula 3 Cloud Computing. Packt Publishing Ltd, 2012.

[52] S. Tu, M. F. Kaashoek, S. Madden, and N. Zeldovich, "Processing analytical queries over encrypted data," in Proceedings of the 39th international conference on Very Large Data Bases. VLDB Endowment, 2013, pp. 289–300.

[53] X. Huang and X. Du, "Efficiently secure data privacy on hybrid cloud," in Communications (ICC), 2013 IEEE International Conference on. IEEE, 2013, pp. 1936–1940.

[54] Z. Zhou, H. Zhang, X. Du, P. Li, and X. Yu, "Prometheus: Privacy-aware data retrieval on hybrid cloud," in INFOCOM, 2013 Proceedings IEEE, 2013, pp. 2643–2651.

[55] CryptDB. Website impact section. [Online]. Available: http://css.csail.mit.edu/cryptdb/#Impact [retrieved: November, 2014]

[56] S. Kamara, C. Papamanthou, and T. Roeder, "Cs2: A searchable cryptographic cloud storage system," Microsoft Research, TechReport MSR-TR-2011-58, 2011.

[57] Kerberos. Keberos documentation. [Online]. Available: http://tools.ietf.org/html/rfc4120 [retrieved: November, 2014]